

Object-Oriented Programming Language

04/13/2023

Due:04/17/2023 11:59p.m.

Lab Assignment 7

1. Please write a program that asks the user to input a sequence of positive integers, terminated by a negative integer. Your program should **dynamically allocate an array** to hold the integers, and then store the integers in the array. After the user has finished inputting the sequence of integers, your program should compute and display the sum of the integers.

Input Specification:

The input contains a sequence of integers, separated by spaces, terminated by a negative integer. The sequence contains at most 100 integers.

Output Specification:

Your program should output a single integer, the sum of the integers in the sequence.

Sample runs of the program are as follows:

```
3 4 45 2 9 -3
63
```

2. You are given two matrices A and B, each of size n by m, and you are to compute their product $C = A * B$. Write a C++ program to perform this calculation using **dynamic memory allocation**.

Input Specification:

The input consists of the following:

The first line contains two integers n and m, the dimensions of matrices A and B.

The next n lines each contain m integers, the entries of matrix A.

The next m lines each contain n integers, the entries of matrix B.

Output Specification:

The output should be a matrix C of size n by n, representing the product of A and B.

Sample runs of the program is as follows:

```

2 3
1 2 3
3 4 5
4 5
6 7
8 9
40 46
76 88

```

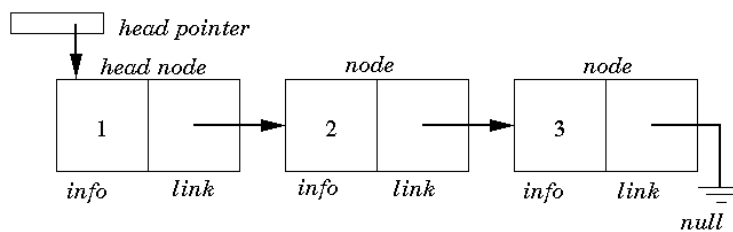
```

2 2
1 1
-1 1
0 1
1 0
1 1
1 1
1 -1

```

3. Please write a program that uses the `shared_ptr` to implement a simple linked list structure with size 3 and print out the linked list.

A linked list stores data and a pointer to the next element. The last element stores a pointer to nullptr.



A Linked List

Here is the code you shouldn't change:

```

struct Node {
    int value;
    shared_ptr<Node> next;
};

int main() {
    // Please fill this blank to establish the linked list

```

```

    shared_ptr<Node> curr = head;
    while (curr) {
        //Please fill this blank (this while loop prints out the
list)
    }

    return 0;
}

```

Input Specification:

The input contains 3 integers, separated by spaces.

Output Specification:

Your program should output 3 integers

Sample runs of the program is as follows:

```

2 3 4
2 3 4

```

4. Please using the following declarations to complete this problem.

```

struct IntArray{
    int* ia; // pointer to first element of int array
    size_t n; // size of the array
};

IntArray creatIntArray();
int findMax(const IntArray&);
void printIntArray(const IntArray&);
void deleteIntArray(IntArray&);

```

In this problem, the IntArray structure is used as the data to be passed around functions.

The first function requires the dynamic allocation of an array of integers for the ia member of the IntArray structure. You will use the new keyword for the memory allocation. To prevent memory leak, you will deallocate the array in the function deleteIntArray(IntArray&), using the delete keyword.

The following is the main method you cannot change.

```

int main(){
    IntArray intArray = creatIntArray();

```

```

    int i = findMax(intArray);
    cout << "Max value in integer array is: " << intArray.ia[i]
<<endl;
    printIntArray(intArray);
    deleteIntArray(intArray);
    return 0;
}

```

Example runs:

```

How many integers do you want to input: 5
Please input the integers: 1 2 3 4 5
Max value in integer array is: 5
Integer Array: 1 2 3 4 5

```

```

How many integers do you want to input: 3
Please input the integers: 1 3 2
Max value in integer array is: 3
Integer Array: 1 3 2

```

5. Create a class OOPClass, with the following details:

- count: static int, which counts how many objects have been created.
- printCount(): static void, prints the current count value.

You will need to also have the appropriate constructors. In this problem, please make data members all private and member functions all public.

The following is the main program you cannot change:

```

int main(){
    OOPClass a1;
    OOPClass a2;
    OOPClass a3;
    OOPClass::printCount();
    OOPClass a4;
    OOPClass a5;
    OOPClass::printCount();
}

```

Output should be look like this:

```

Instance of OOPClass created
Instance of OOPClass created
Instance of OOPClass created
Instance of OOPClass: 3
Instance of OOPClass created
Instance of OOPClass created
Instance of OOPClass: 5

```

6. Suppose you have a class called `Person` which has a private string member variable `name` and a public member function `void introduce() const` that will output "Hi, my name is [name]".

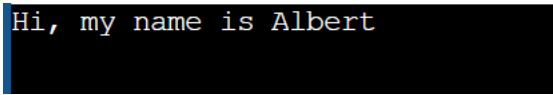
Write a function `std::unique_ptr<Person> createPerson(const std::string& name)` that returns a `unique_ptr` pointing to a `Person` object with the name passed as a string parameter to the function.

In the main function, use the `createPerson` function to create a `unique_ptr` object named `p` that contains a `Person` object with the name "John" and call the `introduce` function of the `Person` object.

The following is the main program you cannot change:

```
int main() {
    unique_ptr<Person> p = createPerson("Albert");
    p->introduce();

    return 0;
}
```



Hi, my name is Albert

7. Write a function that returns the number of times it has been called. For example, the first time the function is called, it should return 1; the second time, it should return 2, and so on. Use a static variable to implement this functionality and ensure that the variable remains unchanged between function calls.

The following is the main program you cannot change:

```
int call_count() {
    //Please fill this blank
}

int main() {
    cout << "Call count: " << call_count() << endl;
    cout << "Call count: " << call_count() << endl;
    cout << "Call count: " << call_count() << endl;
    cout << "Call count: " << call_count() << endl;
    return 0;
}
```

```
Call count: 1  
Call count: 2  
Call count: 3  
Call count: 4
```

8. You can find the `class_initilization.cpp` file in today's module. Please think about the difference of function `m()` and `m2()`. The key is using default or value initialization when creating an object, without user defined constructor. Why is the behavior different? You can refer to this page to seek for an explanation:

https://en.cppreference.com/w/cpp/language/value_initialization